Proyecto para el No Entrenamiento de Imágenes

repositorio: https://git-codecommit.us-east-1.amazonaws.com/v1/repos/api-notrainin

Instancia GCP

Proyecto	*Inference*
Nombre de la máquina	*Máquina de no reconocimiento 0001*
Nombre de la instancia	*Instance-no-training-01*
Zona	*us-east1-b*
Uso	*No training used(True)*
Ip	*Dinámica (Asignada al momento de encender la máquina)*

Sistema Operativo	*Ubuntu*
Dependencias fundamentales del sistema	• *Python3* • *python-dev* • *python3-pip* • *python3-dev* • *libpq-dev* • *nginx* • *curl* • git • *build-essential* • *libssl-dev* • *libffi-dev* • *redis-server* • *libxml2-dev* • *libxslt1-dev* • *zlib1g-dev* • *rabbitmq-server*
Dependencias Fundamentales del entorno	• *Django 3.1.7* • *Celery and Rabbitmq* • *Gunicorn 20.0.4* • djangorestframework 3.12.2 • requests 2.22.0

Directorio raíz del Proyecto:
/home/no_training/api-notraining/api_connect

Directorio de archivos binarios
/home/no_training/api-notraining/bin

En el directorio a su vez podrá encontrar dos folders o carpetas (nginx y service) además de tres archivos; clone_repo.py, sending_ip.py los cuales son scripts python y un último archivo el cual es un bash de linux.

- ***nginx:*** *La carpeta contiene (01) archivo de configuración de nginx (api_conect_nginx) el cual debe ser copiado en el directorio de sitios habilitados.*
- ***service:*** *La carpeta contiene (05) archivos con los servicios que se deben aplicar al momento de encender la máquina.*
  - ***celery.service*** *(activa el celery en el equipo)*
  - ***gunicorn.service*** *(activa gunicorn en el equipo)*
  - ***repositorie.service*** *(clona el repositorio al momento de encender el equipo)*
  - ***rw_nginx.service*** *(sobreescribe el archivo de nginx)*
  - ***sendingip.service*** *(envia la ip y el nombre de la máquina al momento de encender el equipo)*
- ***clone_repo.py:*** *Este scrip python sencillamente se encarga de gestionar todo lo referente al repositorio, el elimina el directorio, lo crea y clona el proyecto cada vez que se enciende el equipo, tiene permisos de lectura y escritura, se ejecuta con el servicio repositorie.service*
- ***sending_ip.py:*** *Este script python envia la dirección ip y el nombre de equipo hacia saturno con el fin unico de dar inicio al envio de información requerida para el procesamiento de las imágenes para el módulo NoTraining. Tiene permisos de lectura y escritura, se ejecuta con el servicio sendingip.service.*
- ***replace_nginx.conf.sh:*** *Este es un script de ejecución o bash de linux para realizar la reescritura del archivo de configuración del app en gnix, se ejecuta con el servicio rw_nginx.service*

el proyecto cuenta con 3 folders o carpetas, una base de datos y un par de archivos, uno de extensión python (.py) y otro archivo de texto plano (.txt).

- ***Config:*** *En este directorio se encuentran todos los archivos de configuración general del proyecto.*
  - ***Celery.py***
  - ***settings.py***
  - ***urls.py***

    - ***asgi.py***
    - ***wsgi.py***
- ***app:*** *Esta es la única aplicación del proyecto y en ella se encuentran con archivos importantes en donde se desarrolla el contenido principal de la app.*
    - ***Api.py:*** *se encuentra la clase* ***ReceiverNoTrainningViewset*** *la cual define el api que recibe la información para el posterior procesamiento de los datos, esta clase se realizón con un* ***ViewSet*** *de Django Rest Framework y se accede a ella únicamente pasando un token en el header de la petición desde el propio sistema y de cualquiera que posea dicho token.*
    - **permissions.py:** Acá se encuentra la clase que define el permiso para acceder a las apis que tengan declarado un atributo permission_class igual a una lista de permisos en la cual la clase **IsSystemInternalPermission** forme parte.
    - ***tasks.py:*** *acá estarían declaradas todas las tareas asíncronas para ser ejecutadas con celery*
    - ***urls.py:*** *Acá declaramos las rutas propias de la aplicación*

ReceiverNoTrainningViewset: *Este es un ViewSet que se encarga de recibir via post un diccionario de datos con las claves country, channel, label, category, zip_art, arts_path, no_transaction_id y ntd_transaction_id, todos estos provenientes de saturno.*

- ***Country****: hace referencia al país seleccionado al momento de crear el paquete.*
- ***Channel****: hace referencia al canal seleccionado al momento de crear el paquete.*
- ***Category****: Esta categoría de manera predeterminada es POP y es seleccionada en el backend al momento de generar un nuevo paquete.*
- ***Label****: hace referencia a los labels del país y canal seleccionados al momento de crear un paquete pero este es seleccionado en la creación del detalle de ese paquete.*
- ***zip_art****: hace referencia a la url del archivos.*
- ***arts_path****: hace referencia al listado de url de cada imagen contenida en el zip.*
- ***no_transaction_id:*** *hace referencia al id único que del paquete.*
- ***ntd_transaction_id****: hace referencia al id único del detalle de ese paquete, este es utilizado para cambiar de estado el registro en saturno y luego para apagar la instancia de la maquina asociada a ese registro.*

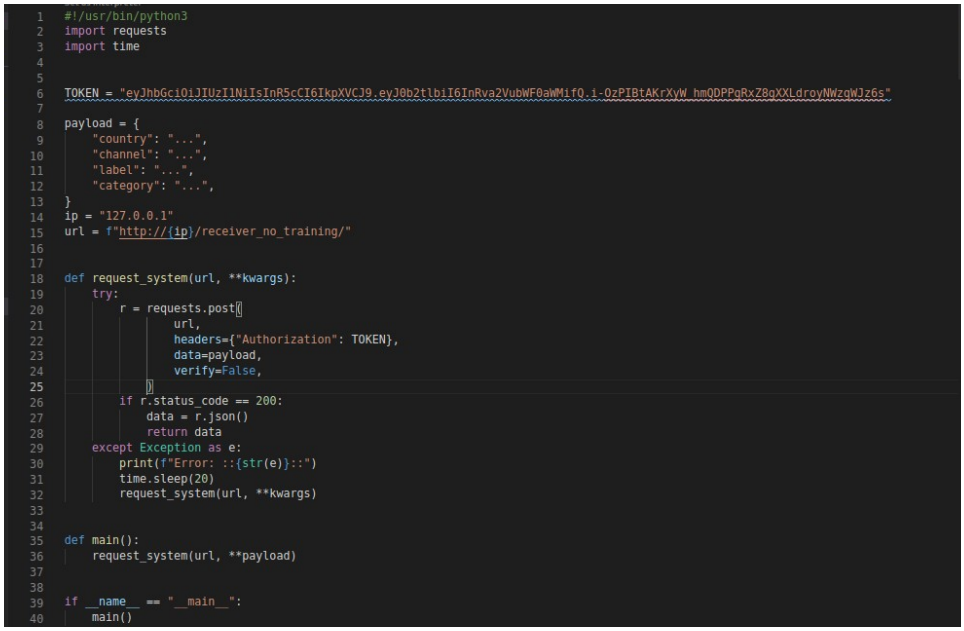
Es importante mencionar que el proyecto tiene solo una terna de rutas, la de el admin, la de solicitud de token y la última para la recepción de datos provenientes de saturno para iniciar el no procesamiento.

De momento la ruta para la solicitud del token de autenticción no se esta utilizando pues para acceder a las apis existentes y futuras se ideo un permiso el cual esta descrito anteriormente y que esta definido en el script con ruta /home/no_training/api-notraining/api_connect/app/perimissions.py.

Importante:

TOKEN = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbiI6InRva2VubWF0aWMifQ.i-OzPIBtAKrXyW_hmQDPPgRxZ8gXXLdroyNWzqWJz6s'

Ejemplo de petición al sistema



```python
#!/usr/bin/python3
import requests
import time


TOKEN = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbiI6InRva2VubWF0aWMifQ.i-OzPIBtAKrXyW_hmQDPPgRxZ8gXXLdroyNWzqWJz6s"

payload = {
    "country": "...",
    "channel": "...",
    "label": "...",
    "category": "...",
}
ip = "127.0.0.1"
url = f"http://{ip}/receiver_no_training/"


def request_system(url, **kwargs):
    try:
        r = requests.post(
                url,
                headers={"Authorization": TOKEN},
                data=payload,
                verify=False,
            )
        if r.status_code == 200:
            data = r.json()
            return data
    except Exception as e:
        print(f"Error: ::{str(e)}::")
        time.sleep(20)
        request_system(url, **kwargs)


def main():
    request_system(url, **payload)


if __name__ == "__main__":
    main()
```

En celery.py esta definida una tarea asíncrona con nombre no_trainning_data la cual recibe un diccionario que debe contener de manera obligatoria la clave ntd_transaction_id, el objetivo especifico de la tarea es enviar esa clave del diccionario indicando el fin del proceso para que sea saturno el encargado de apagar la instancia de la máquina en donde se esta ejecutando el proceso de No entrenamiento y que a su vez sirve de alojamiento para esta pequeño sistema.